sinclair

SPECTRUM

ZX

INTRODUCTION

sinclair

ZX SPECTRUM

INTRODUCTION

by Steven Vickers
and Robin Bradbeer

First Edition 1982

© 1982 by Sinclair Research Limited

Front cover illustration by John Harris of Young Artists

Contents

CHAPTER 1 **The computer and setting it up.** *Page 5*

CHAPTER 2 **The keyboard.** *Page 8*

CHAPTER 3 **Numbers, letters and the computer as a calculator.** *Page 11*

CHAPTER 4 **Some simple commands.** *Page 14*

CHAPTER 5 **Simple programming.** *Page 17*

CHAPTER 6 **Using the cassette recorder.** *Page 21*

CHAPTER 7 **Colours.** *Page 25*

CHAPTER 8 **Sound.** *Page 27*

CHAPTER 9 **Inside the case.** *Page 29*

1. The Computer and setting it up

This short booklet has been written for two types of people. First, those who know nothing, or next to nothing, about computers, and, secondly, for those who are familiar with computer based systems but who like to read instruction booklets before plugging anything in.

There is a second, thicker book which is the BASIC programming manual. This should not be read by the novice computer user until this booklet has been read and understood.

Unpacking the ZX Spectrum, you will have found:

1 This introductory booklet and the BASIC programming manual.

2 The computer. This has three jack sockets (marked 9V DC IN, EAR and MIC), one TV socket, and an edge connector on the back where you can plug in extra equipment. It has no switches – to turn it on you just connnect it to the power supply.

3 A power supply. This converts mains electricity into the form that the ZX Spectrum uses. If you want to use your own power supply, it should give 9 volts DC at 1.4 A unregulated.

4 An aerial lead about 2 metres long, which connects the computer to a television.

5 A pair of leads about 75 cms long with 3.5 mm jack plugs at each end. These connect the computer to a cassette recorder.

You will also need a television – the ZX Spectrum can work without one, but you won't be able to see what it is doing! It must be a UHF television (in the UK); if it is not built to receive BBC2 then it is no good. As its name implies, the ZX Spectrum gives a colour signal which, if you have a colour television, will produce a colour picture. If you only have a black and white television, then the colour will appear as black, white and six different shades of grey; but apart from that, a black and white television will work just as well as a colour television.

The components of the system should now be interconnected thus:

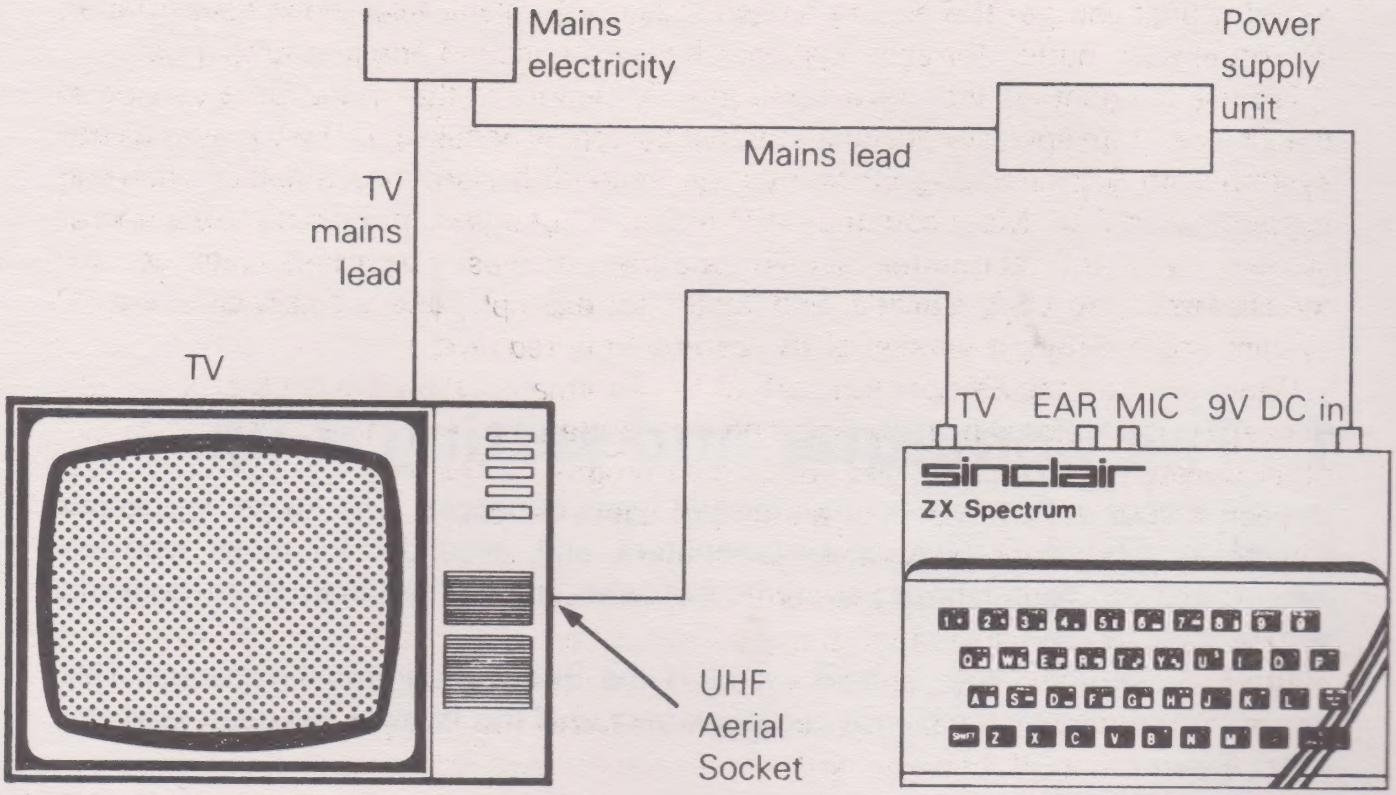

Figure 1

If your television has two aerial sockets, marked UHF and VHF, then use the UHF one (UK).

Turn the power on and switch on the television. You now need to tune the television in. The ZX Spectrum operates on channel 36 UHF (UK), and when it is first plugged in and properly tuned it gives a picture like this:

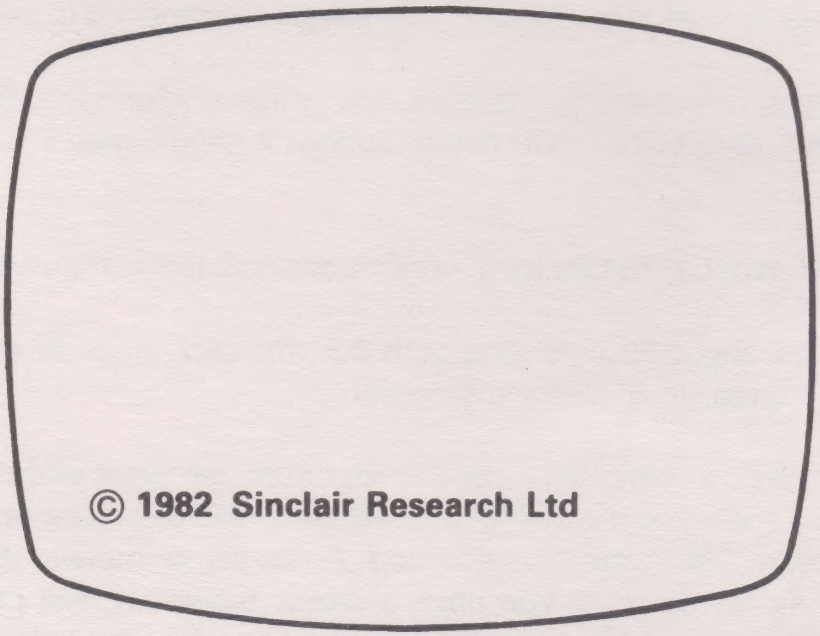

Figure 2

When using the computer, you will probably want to turn the volume on the television right down.

If your television has a continuously variable tuning control, then you just have to

adjust it until you get the picture shown in figure 2. Many televisions now have an individual push button for each station. Choose an unused one and tune it in.

For use in countries that have a different TV system to that in the UK a version of the ZX Spectrum specially designed for that system is necessary. The UK uses a UHF system with 625 lines and 50 frames per second. It also uses a colour encoding system called PAL. Most countries in Western Europe (except France) use a similar system, and the computer should operate in these countries without any modification. The USA, Canada, and Japan, for example, use a totally different TV system and a different version of the computer is required.

When you turn the ZX Spectrum off, all the information stored in it is lost. One way of keeping it for later is by recording it on a cassette tape. You can also buy tapes that other people have prepared and so run their programs. The lead with two jack plugs at each end is used to connect a standard cassette recorder to the ZX Spectrum. Chapter 8 of this booklet explains this further.

Now that you have set up the computer, you will want to use it. The rest of this booklet tells you how to do that; but in your impatience you will probably already have started pressing the keys on the keyboard, and discovered that this removes the copyright message. This is good; **you cannot harm the computer in this way.** Be bold. Experiment. If you get stuck, remember that you can always reset the computer to the original picture with the copyright message by taking out the '9V DC IN' plug and putting it back again. This should be the last resort because you lose all the information in the computer.

WARNING. Do not try to use the ZX 16K RAM with the ZX Spectrum. It will not work.

2. The Keyboard

The keyboard of the Spectrum is very similar to a standard typewriter. The letter and number keys are in the same place; however each key can perform more than one function. On a normal typewriter the letters appear in lower case, and when used in conjunction with the shift key, appear as upper case (capitals). The Spectrum keyboard is just the same.

To help you know what mode the keyboard is in, a reversed out (white on black) letter appears on the screen, indicating the position of the next character that appears when a key is pressed. The letter is flashing to distinguish it from any character already on the screen. It is called the cursor.

When first switched on the Spectrum shows a copyright message on the screen. Pressing any key brings up the word printed below the letter on the key, (this is called the keyword). This is because the computer is expecting a command from you to tell it what to do and all commands *must* begin with a keyword. Unlike most other computers the Spectrum allows you to enter keywords with only one key depression.

For example, if the **P** key is pressed immediately after turning on, the keyword **PRINT** appears on the screen. The **"** symbol is marked on the **P** key as well. To get it, you must press two keys at once; hold down the **SYMBOL SHIFT** key, which is near the bottom right-hand corner of the keyboard, and while still doing that, press the **P** key.

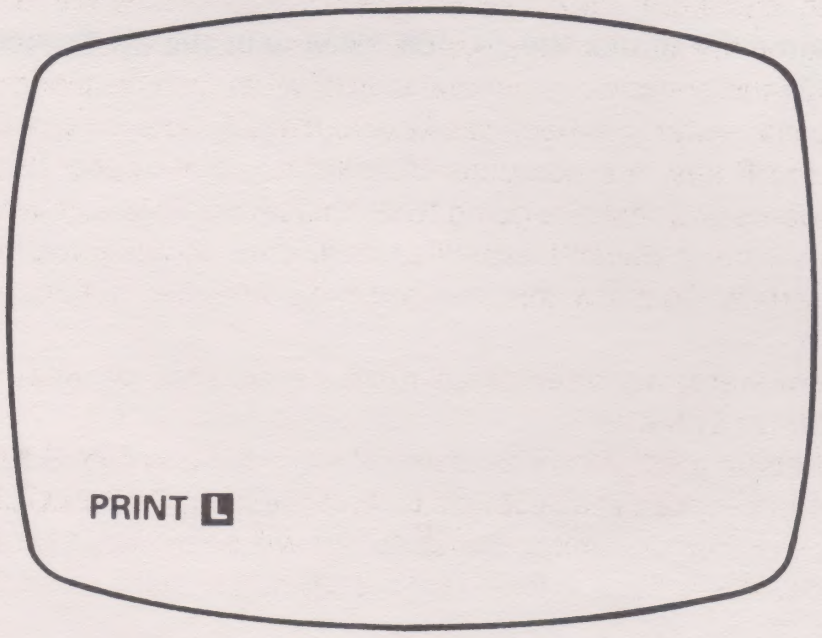

Figure 3

The cursor now changes to an **L**, as a letter is now expected by the computer. Type in the letters **"Hello"**. If there is already some other text, for example, on the screen turn the computer off (remove the 9V plug) and start again. Use the **CAPS SHIFT** key to get the upper case **H**. In general, anything coloured white above the key requires **CAPS SHIFT** to access, and anything coloured red on the key requires **SYMBOLS SHIFT**.

A command beginning with **PRINT** tells the computer to write the letters enclosed

in the double quotation marks onto the screen. For this command to be executed by the computer, the **ENTER** key must be used. When this has been done the screen should display the word

Hello

and some other characters. (A flashing question mark indicates a mistake somewhere. If this happens start again and repeat the exercise). The message at the bottom is really the computer reporting back that everything has gone 'OK'. The message is important when running programs but can be ignored at the moment.

Notice something else: The letter O and the numeral 0 are represented by different characters. It is important to remember this. The numeral 0 *always* has a line through it. The computer will *always* interpret the letter O as a letter, so don't press the wrong key. Similarly, the numeral 1 and the lower case letter L are different, and unlike some typewriters, these cannot be interchanged.

As the keyboard mode is so important to understand it is useful to summarise what happens once again.

The flashing character **L** is called the cursor. It shows whereabouts on the screen the computer will put the next thing that you type. It is not always an **L**; if you turn the computer off and on and then press **ENTER**, the copyright message will change into a **K** cursor. The letter that it uses tells you how the computer will interpret the next thing that you type. At the beginning of a line, it will be a flashing **K** standing for 'keyword'. (The copyright message and reports also count as a flashing **K**). A keyword is one of the computer's special words, occurring at the beginning of a command to give the computer a general idea of what the command is going to tell it to do. Since the computer is expecting a keyword at the beginning of a line, when you press – say – the **P** key, the computer decides not to interpret this as a **P**, but as **PRINT**; and it warns you that it is going to do this by making the cursor a **K**. When it has the first keyword, it doesn't expect another one, so what you type now will be interpreted as letters. To show this, the computer changes the cursor to an **L** – for 'letter'.

These different states are often called *modes* – we shall talk about keyword (or K) mode, and letter (or L) mode.

If you want to type a lot of capital letters without holding **CAPS SHIFT** down, you can make all letters come out as capitals by first pressing **CAPS LOCK** (**CAPS SHIFT** with **2**). To show this is happening, the **L** cursor will be replaced by a flashing **C** (for 'capitals'). To get lower case letters and the **L** cursor back, press **CAPS LOCK** a second time.

(If you press **CAPS LOCK** during keyword mode, you will not immediately notice any difference, but you will see the effect after entering the keyword when the computer will be in C mode instead of L mode).

As well as keywords, letters, numbers and various programming and scientific expressions, the keyboard also has eight graphics characters. These appear on the number keys **1** to **8**, and can be printed onto the screen in a similar way to letters and numbers. To do this the keyboard must be changed to *graphics mode*. This is done by

pressing the **CAPS SHIFT** key with the **9**. Notice the cursor change to a **G**. Pressing the **9** key will change back to L mode.

There is one last mode that the keyboard can be changed to. The *extended mode*, indicated by an **E** cursor, is obtained by pressing **CAPS SHIFT** and **SYMBOL SHIFT** at the same time. This allows most of the scientific and programming functions to be used. Pressing the two **SHIFT** keys again will revert the keyboard back to letter, L, mode.

Even if you are the most proficient typist, or programmer, wrong keys get pressed. So far the only way to overcome this has been to pull the plug out! Although this may be convenient if only one command has been given to the computer, it is certainly very inconvenient if a lot of information has already been entered.

Luckily we can use the **DELETE** key to change errors. For example, not much can go wrong with the simple command:

PRINT "Hello"

. . . or can it?

Let's assume that you didn't use the **SYMBOL SHIFT** key to get the opening quotes. The screen would show

PRINT PHello"

The computer would not have recognised what came after **PRINT** as no quotes indicates to the ZX Spectrum that a number is expected – and it found a letter instead. It shows its confusion by flashing a **?** at the end of the line.

Fortunately you don't have to type it all out again. On the top row of the keyboard are four arrows pointing in different directions and the word **DELETE**. To operate these keys, you have to use the **CAPS SHIFT** key when you press them. The sideways arrows move the cursor to the left or the right, and the **DELETE** key rubs out the character immediately before the cursor.

To correct your nonsense line, press ← (**CAPS SHIFT** and **5** at the same time) until the cursor is just after the **P** that you put in by mistake – if you hold them down for a second or two, then they will start working continuously, emitting a quiet clicking sound. In fact, if you keep your finger on any key for more than about three seconds it automatically repeats itself. Key in **DELETE** (**CAPS SHIFT** and **0**) to delete the wrong **P**, and then type **"** (**SYMBOL SHIFT** and **P**) to insert what should have been there – note that it is inserted without overwriting anything else. Try the cursor right key as well, just to get the hang of it. If you made any genuine typing mistakes, correct them in the same way, remembering that you can't overwrite mistakes; you have to delete them and then insert the corrections.

Now, when you press **ENTER**, the computer will write your message at the top of the screen – or underneath the one it did the first time, if that is still there.

A full description of the keyboard can be found in Chapter 1 of the BASIC programming manual.

3. Numbers, letters and the computer as a calculator

We have already seen how to tell the computer to print letters and graphics on the screen using **PRINT**. We have also seen that **ENTER** has to be used to tell the computer to execute the command just typed in. From now on, we will not use **ENTER** in the manual each time a command is used, but assume that you will key it at the end of the line automatically.

Numbers can be handled by the computer more easily than letters. In the previous chapter we hinted at this by explaining that the computer expects a number after **PRINT** if quotations marks are not used.

So if we key

PRINT 2

the number **2** will appear on the screen.

It is possible to mix letters and numbers:

PRINT 2,"ABC"

Notice that there is a gap on the screen between the **2** and **ABC**.

Now key

PRINT 2;"ABC"

and then

PRINT 2 "ABC"

Using a comma between the items after **PRINT** spaces them out by *16* columns, using a semi-colon leaves no spaces and using nothing gives an error.

PRINT can also be used with the mathematical functions on the keyboard. In fact the ZX Spectrum can be used as an electronic calculator.

For example:

PRINT 2+2

The answer appears at the top of the screen. Compare this with:

PRINT "2+2"

It is possible to combine these to give something more useful. Try

PRINT "2+2=";2+2

Try some other kinds of arithmetic as well:

PRINT 3–2
PRINT 4/5
PRINT 12*2

The * is used as a multiplication sign instead of X to avoid getting it confused with the letter **x**; and **/** is used for the division sign ÷.

Experiment with lots of different calculations. If you like you can use negative numbers or numbers with decimal points in.

If you do enough to use up the 22 lines of the top part of the screen, then you will notice something rather interesting happening; it will all move up one line and the top line will be lost. This is called *scrolling*.

Calculations are not always performed in the order you might expect. As an example, try

PRINT 2+3*5

You might expect this to take 2, add 3, giving 5, and then multiply by 5, giving 25; however this is not the case. Multiplications – and divisions as well – are performed before additions and subtractions, so the expression '2+3*5' means 'take 3 and multiply it by 5, giving 15; and then add that to 2, giving 17'. 17 should be the answer displayed on the screen.

Because mulitplications and divisions are done first, we say that they have higher priority than addition and subtraction. Relative to each other, multiplication and division have the same priority, which means that the multiplications and divisions are done in order from left to right. When they are dealt with, we are left with the additions and subtractions – these again have the same priority as each other, so we do them in order from left to right.

Let us see how the computer would work out:

PRINT 20–2*9+4/2*3

i 20–2*9+4/2*3
ii 20–18+4/2*3
iii 20–18+2*3

First we do the multiplications and divisions in order from left to right

iv 20–18+6
v 2+6
vi 8

and then the additions and subtractions

Although all you really need to know is whether one operation has a higher or lower priority than another, the computer does this by having a number between 1 and 16 to represent the priority of each operation: * and / have priority 8, and + and − have priority 6.

This order of calculation is absolutely rigid, but you can circumvent it by using brackets; anything in brackets is evaluated first and then treated as a single number, so that

PRINT 3*2+2

gives the answer 6+2=8 but

PRINT 3*(2+2)

gives the answer 3*4=12

It is sometimes useful to give the computer expressions such as this because whenever the computer is expecting a number from you, you can give it an expression instead and it will work out the answer. The exceptions to this rule are so few that they will be stated explicitly in every case.

You can write numbers with decimal points (use the full stop), and you can also use scientific notation – as is quite common on pocket calculators. In this, after an ordinary number (with or without a decimal point), you write an exponent part consisting of the letter **e**, then maybe −, and, then a number. The exponent part shifts the decimal point along to the right (or left, for a negative exponent), thus multiplying (or dividing) the original number by 10 a few times. For instance,

2.34e0=2.34
2.34e3=2340
2.34e−2=0.0234 and so on

(Try printing these out on the computer.) This is one of the few cases where you can't replace a number by an expression: for instance, you cannot write

(1.34+1)e(6/2).

You can also have expressions whose values are not numbers, but strings of letters. You have seen the simplest form of this many times, the string of letters written down with double quotes around it. This is rather analogous to the simplest form of numeric expression, which is just a number written down on its own. What you have not yet seen is the use of + with strings (but not −, * or /, so there is no problem with priorities here). Adding strings together just joins them together one after the other: so try

PRINT "jers"+"ey cow"

You can add together as many strings as you like in a single expression, and if you want, you can even use brackets.

4. Some simple commands

The computer memory can be used to store all sorts of things. We have seen, so far, that the **PRINT** command allows us to show letters, numbers and the results of calculations using both letters and numbers, on the screen.

If we want to tell the computer to remember a number, or a string of letters, then we have to allocate some of the memory for that use.

Most pocket calculators have a key called 'memory' which is used for remembering numbers for later. Your computer can do much better than that: it can have as many of these imaginary boxes in it as you like, and you write a name on each one.

As an example, suppose you want to remember your age! The **LET** command is used (**LET** is the keyword on the **L** key): let's say it is 34

```
LET age=34
```

What happens when the **LET** command is used, is that a certain section of memory is designated 'age' and the number 34 is stored in it. To get this stored information out type

```
PRINT age
```

and back comes the number 34. It is very easy to change the contents of the 'box' called 'age'. Type:

```
LET age=56
```

then type:

```
PRINT age
```

and 56 should appear on the screen. 'age' is an example of a *variable* so called because its value may vary. It is possible to combine printing a message direct to the screen, and the value of a variable. Type

```
PRINT "Your age is  "; age
```

However the computer is a lot more useful than just remembering numbers with names attached to them. It can also remember strings of letters. To differentiate between number variables and string variables – as they are called – the dollar symbol – **$** – is used at the end of the variable name.

For example: if we wanted to save the string of letters

```
"Your age is"
```

we could call it

a$

(string variable names can only have a single letter, other than the **$**, in them). So type

LET a$="Your age is "

If you now key

PRINT a$

back comes the string of letters on the screen.

If the computer hasn't been turned off since the start of this chapter type

PRINT a$;age

and see what happens.

There are other ways of getting information into the computer's memory without using the **LET** command.

For example the **INPUT** command, in its simplest form, tells the computer that some information is expected from the keyboard. Instead of typing **LET** etc. everytime, you can key

INPUT age

Once the **ENTER** key has been pressed a flashing **L** cursor will appear on the screen. This means that the computer wants some information from you. So type your age and then press the **ENTER** key. Although nothing seems to have happened the variable has now been given the value you typed in. Typing

PRINT age

should prove this.

Let's combine all this together into a series of commands.
Type

```
LET b$="What is your age?"
LET a$="Your age is "
INPUT (b$);age: PRINT a$;age
```

Note that the last line consists of two commands separated by a colon.

INPUT (b$);age

is another way of entering

INPUT "What is your age?"; age

5. Simple programming

Up to now we have been telling the computer what to do, directly from the keyboard. Although it is possible to combine commands together, only limited applications are feasible using this method.

The great thing about computers is that they are programmable. This means that we can give them a series of instructions to make them do things in a sequence.

Every computer has its own language which allows us to communicate with it. Some languages are very simple – so that the computer can understand them easily. Unfortunately languages that are simple for the computer to understand are difficult for humans. In some ways the reverse is also true – languages simple enough for us to understand are relatively difficult for the computer – and even have to be translated or interpreted.

The ZX Spectrum uses a high level language called BASIC.

BASIC stands for Beginners All-purpose Symbolic Instruction Code and the language was designed at Dartmouth college in New Hampshire, USA, in 1964. It is very widely used on personal computers, but although it is broadly similar on all of them, there are subtle differences. That is why this manual is written specifically for the ZX Spectrum. But ZX Spectrum BASIC is not too far from a (non-existent) consensus BASIC and so you should not have too much trouble adapting any BASIC program to work on the ZX Spectrum. Unlike other BASICs ZX Spectrum BASIC does not allow the command **LET** to be omitted when values are being assigned to variables.

There is a limit to how many instructions can be stored in the computer. The ZX Spectrum indicates this limit by emitting a buzz.

When programming in BASIC it is necessary to let the computer know the order in which the instructions are to be executed. Hence each line of the sequence of instructions has a number at its beginning. It is normal to start at 10 and to increase this by 10 for each new line. This allows other lines to be inserted if they have been omitted, or the program to be modified.

Let's look at a simple program: Consider the series of commands at the end of the last chapter. If we wanted to repeat the series of commands it would be necessary to enter them each time. A *program* overcomes that necessity.

Type in the following with **ENTER** after each line.

```
10 LET b$="What is your age? "
20 LET a$="Your age is "
30 INPUT (b$);age
40 PRINT a$;age
```

Note that it is not necessary to enter any spaces, except inside quotes.

Nothing will actually happen until we tell the computer to start working on the program. That is done by using **RUN** (the keyword on **R**).

Enter this command and see what happens.

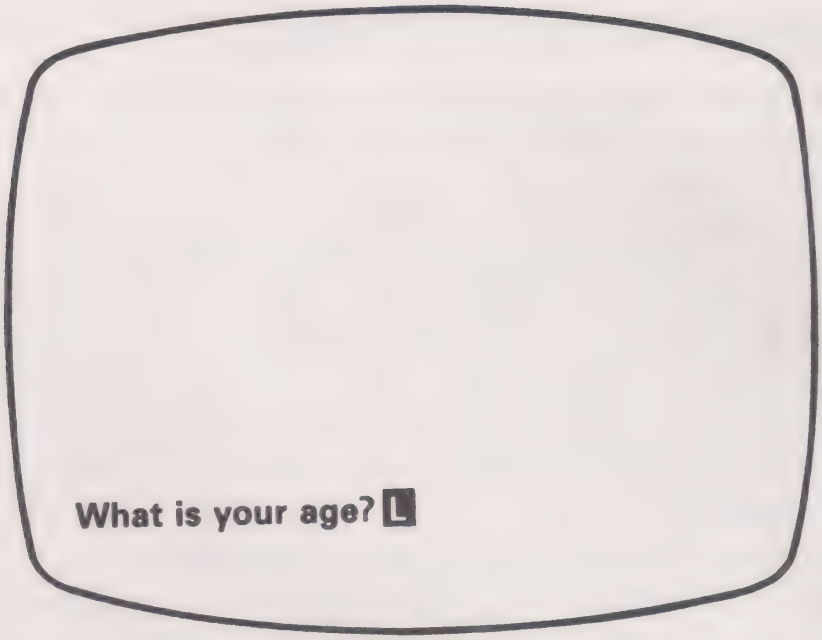

Figure 4

You may also have noticed a right facing arrow when each line has been entered. This indicates the last line entered. If you want to see the program again key **ENTER** again, (or **LIST**). You can use **RUN** to execute the program as many times as you like. When you no longer need this program, you can remove it by using the **NEW** command. This wipes out the program stored in memory, and gives you a 'clean slate' ready to put in a new one.

Key **NEW**, then **LIST** and see what happens.

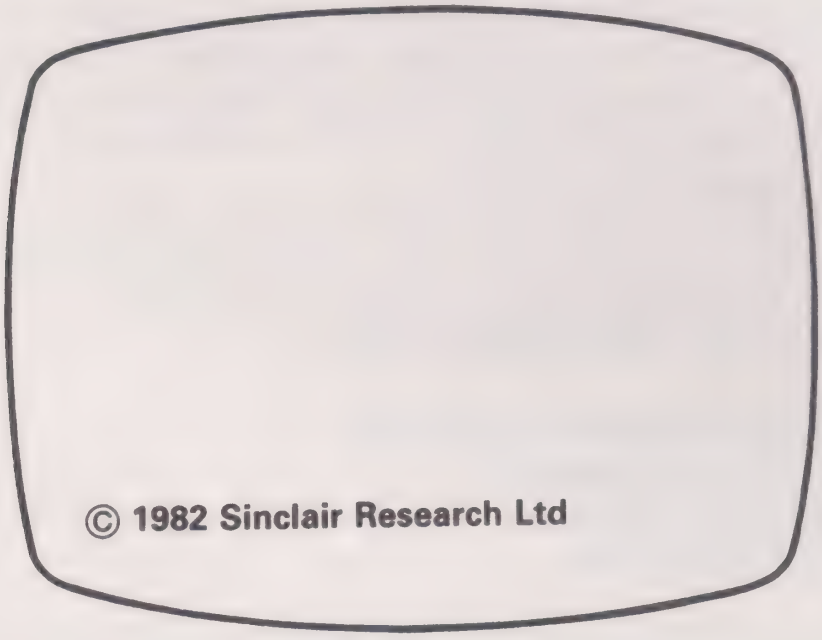

Figure 5

To recap:

When you type in a command preceded by a number, then this tells the computer that it is not simply a command, but a program line. The computer does not execute it, but stores it away for later.

The ZX Spectrum helpfully writes on the screen (or lists) all the program lines that you have entered with a ▶ against the last line that you entered.

The computer will not execute any of these lines immediately, but just stores them away inside itself.

To get the computer to execute these lines, you must use the command **RUN**.

If you press **ENTER** on its own you will get the listing back.

Let's consider another simple program. This one will be a bit more mathematical and print out the squares of all the numbers between 1 and 10 (the square of a number is just that number multiplied by itself).

To generate numbers from 1 to 10 introduces another concept in programming in BASIC. This is the method by which we get the computer to count. Earlier we have seen that numbers can be stored in the computer's memory by attaching a 'name' to them – or technically, assigning a value to a variable. Let the variable **x** start with the value 1 and increase in steps of 1 to 10. This is done by using the command **FOR . . . TO . . . STEP**.

So to enter this program key **NEW** to get rid of the previous one and type the following:

```
10 FOR x=1 TO 10 STEP 1
```

(Normally the **STEP 1** part can be omitted if counting is going up in steps of one).

The next line must now tell the computer what to do with **x** at whatever value it is, so key:

```
20 PRINT x, x*x
```

Finally we need a line to tell the computer to go to the next value of **x**, therefore key

```
30 NEXT x
```

On reaching this instruction the computer goes back to line 10 and repeats the sequence. When **x** exceeds 10 the computer goes to the next line in the program i.e. line 40.

The program should now appear on the screen as follows:

```
10 FOR x=1 TO 10 STEP 1
20 PRINT x, x*x
30 NEXT x
```

For completeness we should really have another line telling the computer that the program has ended when **x**=10 so key

```
40 STOP
```

If the program is now **RUN** two columns should appear, the first with values of **x**, the second with values of **x*x**, or **x** squared. It is possible to label these columns by adding another line, like this

5 PRINT "x","x*x"

Notice that although this has been entered *after* all the other lines, because it has a lower line number the computer automatically puts it in the correct place.

Try writing programs using other mathematical functions. If you have any doubt about how to use them refer to the appropriate pages in the BASIC programming manual.

6. Using the cassette recorder

It is rather tedious having to type programs in to the computer each time you want to use it. The ZX Spectrum has the facility for recording programs onto magnetic tapes with a normal, domestic cassette recorder. If you have a program in the memory try to save it using the following procedure.

If you can save programs on cassette tape you can load them back again later.

Most cassette recorders will work: as far as the computer is concerned, the cheap portable mono cassette recorders are at least as good as expensive stereo ones, and give less trouble as well. You will find a tape counter very useful.

The cassette recorder must have an input socket for use with microphones and an output socket for use with earphones (if there is not one, try the external loudspeaker socket). They should be 3.5 mm jack sockets (i.e. to fit the jack plugs on the leads provided), because other sorts often do not give a signal powerful enough for the computer.

Any cassette tape should work although low noise tapes may be better. Having acquired a suitable cassette recorder, connect it to the computer using the leads supplied with the ZX Spectrum: one lead should connect the microphone input socket on the recorder to the socket marked 'MIC' on the back of the computer and the other should connect the earphone output socket on the recorder to the 'EAR' socket. (You cannot harm the ZX Spectrum by connecting the cables incorrectly).

When you are using the **SAVE** command to store a program onto tape, you must make sure that one of the plugs of the lead connecting the 'EAR' sockets on the computer and cassette recorder is pulled out – either of them will do. If you forget to do this you will get nothing more than a steady note recorded onto the tape, which is useless. The reason for this is that when the cassette recorder is recording it amplifies the signal coming in on its 'MIC' socket, and puts it out on the 'EAR' socket. If this gets back into the computer it will form a loop, which will oscillate, smothering the signal you were trying to record.

Type some program into the computer, say the squares program in the previous chapter, and then type:

SAVE "Squares"

Squares is just a name that you use to label the program while it is on tape. You are allowed up to ten characters in the name which must consist of just letters and numbers.

The computer will have come up with a message **Start tape then press any key**. We shall first go through a dry run so that you can see what happens: do not start the cassette recorder, but press a key on the ZX Spectrum and watch the border of the TV screen. You will see patterns of coloured horizontal stripes.

5 seconds of red and pale blue stripes, about 1 cm wide and moving slowly upwards.

A very short burst of blue and yellow stripes.

1 second with everything as normal.

2 seconds of the red and pale blue pattern again,
about 1 second of the blue and yellow pattern again.

Try it again until you can recognise all these. The information is saved away in two blocks and both blocks have a lead-in corresponding to the red and pale blue pattern, and the information itself, corresponding to the blue and yellow pattern. The first block is a preliminary one containing the name and various other bits of information about the program, and the second is the program itself together with any variables present. The white section between them is just a gap.

Now let's actually capture that signal on cassette tape.

1. Position the tape in a part either that is blank, or that you are prepared to overwrite.

2. Type

SAVE "Squares" (and **ENTER**)

3. Start the cassette recorder recording.

4. Press any key on the ZX Spectrum.

5. Watch the television as before. When the computer has finished (with the report **0 OK**) stop the cassette recorder.

To make sure that this has worked, you can check the signal on the tape against the program in the computer using the **VERIFY** command.

1. Turn the volume control on the cassette recorder to approximately half way and reconnect the 'EAR' lead.

2. Rewind the cassette to somewhere before where you started to record previously.

3. Type

VERIFY "Squares"

(**VERIFY** is extended mode, then shifted **R**).

4. Start the cassette recorder playing.

The television border will alternate between red and pale blue until the tape reaches the recording you made; then you will see the same pattern as you did when you saved the program. In the one second gap in the middle, **Program Squares** will be written on the screen – when the computer is searching for something on tape, it prints up the name of everything it comes across. If you see all this pattern and then the computer stops with report **0 OK** your program is safely recorded on tape and you

can skip the next few paragraphs. Otherwise, something has gone wrong Go through these questions to find out what.

Making sure your program is saved

Has the name come up?

If not then either the program was not saved properly in the first place, or it was, but was not read back properly. You need to find out which. To see if it was saved properly, rewind the tape to just before where you started recording, and play it back through the tape recorder's own loudspeaker (you will probably have to unplug the lead from the earphone socket on the tape recorder). The red and pale blue lead-in gives a very clear, steady high pitched note, and the blue and yellow information part gives a much less pleasant sound, like a morse code message in a hurricane. Both of these are quite loud – at full volume they can easily drown conversation.

If you do not hear these noises then the program probably did not get saved. Check that the right leads are plugged in the right sockets. Make sure that the 'MIC' sockets are connected, and that the 'EAR' sockets are not. It happens with some tape recorders that the jack plug does not make contact if it is pushed right in. Try pulling it out about a tenth of an inch – you can sometimes feel it settling down into a more natural position. Also check that you were not trying to record on the plastic leader at the beginning of the cassette. When you have checked these, try saving again.

If you can hear these sounds as described then **SAVE** was probably all right and your problem is with reading back.

Check the leads again, and also check the volume level. If it is too quiet the computer will not hear the signal properly, and you will not see the right patterns on the screen; if it is too loud the signal will get distorted – you may be able to hear it coming through the computer's own loudspeaker. There is a wide range of acceptable values in between, but you could try experimenting.

The next case is when the computer finds the program and writes its name up but still goes wrong. Some possibilities are:

You mistyped the name, either in **SAVE** (when the computer will write the mistyped name on the screen) or **VERIFY**: the computer will ignore the program and carry on flashing red and pale blue as it goes.

There is a genuine mistake on the tape: the computer will come back with **R Tape loading error**, which means in this case that it failed to verify the program. Save it again.

It is just possible that the volume setting on the tape recorder is not quite right; but it cannot be far wrong because the computer managed to read the first block.

Now let us suppose that you have saved the program and successfully verified it. Loading it back is just like verifying it except that you type

LOAD "Squares"

instead of

VERIFY "Squares"

LOAD is on the **J** key. Since it verified properly, you should have no problem loading.

LOAD deletes the old program (and variables) in the computer before loading in the new one from tape.

Once a program has been loaded, the command **RUN** will run it.

It is possible to buy pre-recorded programs on cassette. They must be specially written for the ZX Spectrum: different types of computer have different ways of recording programs, so they cannot use each other's tapes.

If your tape has more than one program recorded on the same side, then each will have a name. You can choose which program to load in the **LOAD** command: for instance, if the one you want is called 'helicopter' you could type

LOAD "helicopter"

(**LOAD ""** means **LOAD** the first program you come across which can be very useful if you cannot remember the name of your program.)

7. Colours

One of the reasons for buying the ZX Spectrum in the first place was the ability to use colour on the TV screen. The screen is divided into two areas. The outer part is referred to as **BORDER**, the central area as **PAPER**. It is possible to change the colours of these two sections at will, both directly from the keyboard and in a program.

The ZX Spectrum has eight colours to play with, and these are given numbers between 0 and 7. Although the colours look in random order, they do in fact give decreasing shades of grey on a monochrome TV.

Here is a list of them for reference; they are also written over the appropriate number keys:

0 black
1 blue
2 red
3 purple, or magenta
4 green
5 pale blue, or cyan
6 yellow
7 white

When the computer is first switched on the system works in black and white. So the normal value for **BORDER** and **PAPER** is 7 i.e., white. The colour of any character appearing on the screen is defined by the **INK** command. This is normally 0, i.e., black. Initially the three commands controlling the screen colours are set by the computer.

However, you can change these values. For example key

BORDER 2

If you remembered to press the **ENTER** key, the border should now change from white to red. This includes the area at the bottom where commands and instructions are typed in. Try typing in other numbers and see how the colours change.

Now try changing the centre of the screen area by keying

PAPER 5

The **PAPER** command is one of the extended mode commands as mentioned earlier. It is obtained by typing both **CAPS SHIFT** and **SYMBOLS SHIFT** at the same time. **PAPER** is then a shifted **C**. When the **ENTER** key is pressed twice the centre of the screen should change to pale blue. The first **ENTER** cancels the **PAPER** command already stored in the computer, but only when the second **ENTER** is pressed (causing the computer to **LIST** any program and therefore rebuild the screen information) does the new **PAPER** colour get used. If you are using a colour television, and it hasn't

changed colour, try adjusting the colour controls on the television, and maybe the tuning control.

The **INK** command is similar to the **PAPER** command and controls the colour of the characters appearing on the **PAPER** section of the screen. Obviously if the **INK** and **PAPER** colours are the same nothing will appear on the screen!

The **BORDER**, **PAPER** and **INK** commands can be used in programs. Here is a simple one to show the range of colours available:

```
10 FOR x=0 TO 7
20 BORDER x
30 PAPER 7-x: CLS
40 PAUSE 50
50 NEXT x
```

This program, when **RUN**, goes through the eight colours, contrasting the **PAPER** and **BORDER** colours. The **CLS** command after **PAPER** forces the computer to rebuild the screen image and use the new **PAPER** colour. The **PAUSE** command stops the program for 1 second so that we can see what's happening (Try running the program with the **PAUSE** left out.) To show how the **INK** command works type in the following program, after a **NEW** command.

```
10 BORDER 7
20 PAPER 1
30 INK 4
40 PRINT "Green characters on blue background"
```

There are other commands associated with the colour capabilities of the ZX Spectrum, and these are detailed in the BASIC programming manual.

8. Sound

The ZX Spectrum can make sounds of an infinite variety. The frequency of the note and its duration are under the control of the user. The command **BEEP** is used to tell the computer to make a sound. **BEEP** is an extended mode command and is obtained using the **Z** key.

The 'centre' frequency for the **BEEP** command is middle C. This can be varied within the **BEEP** command, and any note can be obtained if it is expressed as semitones or parts of semitones above or below this centre frequency. If the command

```
BEEP 2,0
```

is entered the computer should emit a sound at middle C pitch for two seconds.

The two numbers between them control the sort of note that is emitted, the first giving the length of the note in seconds, and the second the pitch of the note in semitones above middle C. Thus the pitch code for middle C is 0, that for C# is 1, for D it is 2, and so on up to the next C above which is 12, because 12 semitones make one octave. You can carry on to 13 and beyond, if you want, so that the higher the number, the higher the pitch.

Try this:

```
BEEP 1,4: BEEP 1,2: BEEP 2,0
```

You should hear no less than the first bar of Three Blind Mice. Since you can join quite a few **BEEP**s together with colons like this, you could, if you had the patience, produce a whole tune. You might like to try using more notes than just three.

(Colons don't just join **BEEP**s together; you can use them to build composite commands out of any of the elementary commands.)

As a more complicated example, you can make a singing chameleon command by mixing **BEEP** and **BORDER** together:

```
BORDER 1: BEEP 1,14: BORDER 3: BEEP 1,16: BORDER 4: BEEP
1,12: BORDER 6: BEEP 1,0: BORDER 5: BEEP 4,7: BORDER 1
```

(Don't worry about the fact that this stretches over from one line to the next: the computer doesn't take any notice of this.)

A short program to play a whole series of notes could be as follows:

```
10 FOR x=0 TO 24
20 BEEP 2, x
30 NEXT x
```

There are a lot more things that can be done with this command – see the BASIC

programming manual for other ideas.

For notes lower than middle C the number of semitones is indicated by a negative number.

9. What's Inside the Case?

The picture here shows what the inside of the ZX Spectrum looks like.

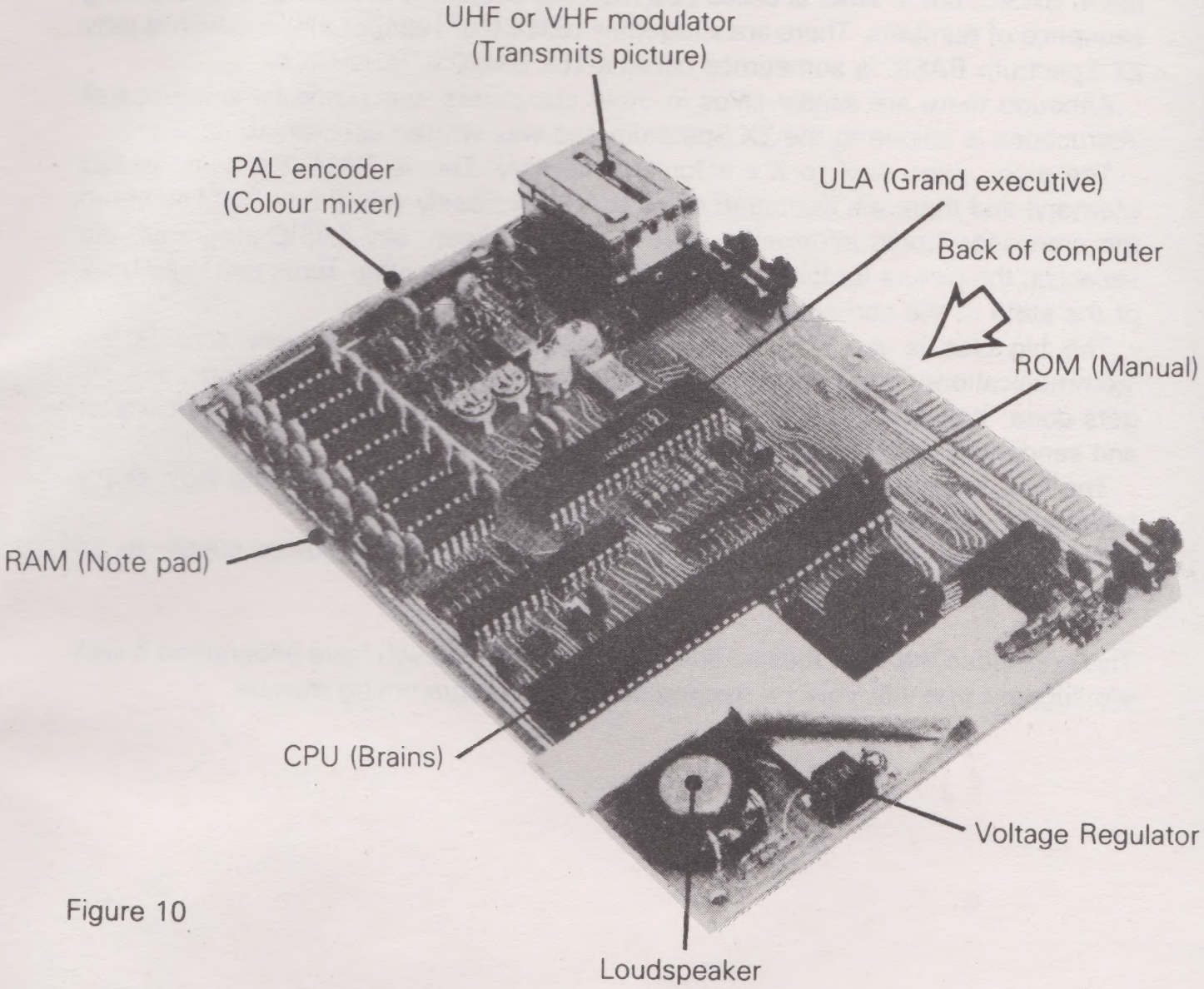

Figure 10

As you can see everything is named by a three letter abbreviation. The black rectangular pieces of plastic with lots of metal legs are the integrated circuits that actually do all the work. Inside each one is a $\frac{1}{4}" \times \frac{1}{4}"$ square of silicon joined by wires to the metal legs. On that silicon chip are thousands of transistors that make up the electronic circuits that *are* the computer.

The brains behind the operation is the processor chip, often called the CPU (Central Processor Unit). This particular one is called a Z80A, which is a faster version of the popular Z80.

The processor controls the computer, does the arithmetic, looks at what keys have been pressed, decides what to do as a result, and in general decides what the computer should do. However, for all its cleverness, it could not do all this on its own. It knows nothing about BASIC or decimal point arithmetic, for example, and it has to

get all its instructions from another chip, the ROM (Read Only Memory). The ROM contains a long list of instructions that make a computer program, telling the processor what to do under all foreseeable circumstances. This program is written not in BASIC, but in what is called Z80 machine code, and takes the form of a long sequence of numbers. There are altogether 16384 (16*1024) of these, which is why ZX Spectrum BASIC is sometimes called a 16K BASIC – 1K is 1024.

Although there are similar chips in other computers, this particular sequence of instructions is unique to the ZX Spectrum and was written specially for it.

The eight chips next to it are for the memory. This is RAM (Random Access Memory) and there are two other chips that work closely with them. RAM is where the processor stores information that it wants to keep, any BASIC programs, the variables, the picture for the television screen and various other items that keep track of the state of the computer.

The big chip is the SCL (Sinclair Computer Logic) chip. It really acts as the 'communications centre', making sure that everything the processor requires actually gets done; it also reads the memory to see what the television picture consists of and sends the appropriate signals to the TV interface.

The PAL encoder is a whole group of components that converts the logic chip's television output into a form suitable for colour televisions.

The regulator converts the slightly erratic voltage of the power supply to an absolutely constant five volts.

This concludes the Introductory Booklet. If you feel that you have understood it well we suggest that you now try reading the BASIC programming manual.

Sinclair Research Limited
25 Willis Road,
Cambridge CB1 2AQ
England